Suppressing Non-Semantic Noise in Masked Image Modeling Representations

Supplementary Material

A. Self-Supervised Objectives

Contrastive Learning. In this work, we focus on the contrastive self-distillation objective as presented in Caron et al. [11]. Given an image x, two random augmentations are applied yielding the views u, v. The views are sent through a teacher-student framework giving $p = f_\theta(v) \in \mathbb{R}^d$ and $q = f_{\hat{\theta}}(u) \in \mathbb{R}^d$, where θ and $\hat{\theta}$ denote the teacher and student weights, respectively. The loss minimizes the cross-entropy

$$\mathcal{L}_{\text{CLS}} = -q^\top \log p. \tag{A.1}$$

Importantly, this loss is applied on the global representations, given by the `CLS` token in ViTs. The teacher and student share the same architecture, comprising a backbone and a projection head for the global representation. The teacher is updated as an exponential moving average of the student.

Masked Image Modeling. The core idea of the MIM objective is to reconstruct masked parts of an image when given visible parts as context. While the masking strategy varies between the MIM-based methods, the general setup can be summarized as follows. The input x is obscured by a random mask m and passed through an encoder to give a context $z = f(m(x))$ with which to make a prediction $\hat{s} = g(z)$ about the unmasked image x. The prediction can be made in the pixel space or in the latent space where the target s is given by passing x through the encoder. The loss is

$$\mathcal{L}_{\text{MIM}} = \ell(\hat{s}, s) \tag{A.2}$$

where ℓ is the mean squared error [22], euclidean distance [2] or negative cross entropy [16, 26, 29, 53]. Notably, several frameworks [26, 29, 53] employ a combination of contrastive loss over the global `CLS`-tokens in a multi-view setup with a local MIM-based loss over masked patch tokens.

B. Details on Activation and Distributions

In this section, we elaborate on the details of responses, activations, and distributions. As explained in the main text, we compute responses for all images in the dataset for each principal component. We binarize the activations, and compute the token-wise empirical distributions as individual Bernoulli distributions for each token in the image, $P_{d,n}$ from Section 3.3. As an ensemble, this can be taken as $P_d \sim \text{Multinomial}(2, N)$, forming a full distribution over the image.

Figure B.1. The relation of soft responses of DINOv2 to PC_1 in Figure 2 (left), compared with the multinomial distribution of activations P_1 from Figure 4 and Section 3.3 (right). The responses are binarized, and the probability distribution P_1 is computed as a multinomial distribution over all tokens in the image.

Divergence measures would be a natural choice to compare P_d, Q_d, however, we find that pure divergence measures such as Jensen-Shannon yield suboptimal scores in our testing. If $P_{d,n} = Q_{d,n} = 0.5$, a proper divergence such as the Jensen-Shannon requires that the distributions are equal, e.g., $D_{\text{JS}}(P_{d,n}, Q_{d,n}) = 1$. However, in this case we are unsure if individual activations actually agree or not. In other words, a proper divergence yields high scores when activation maps are highly uncertain.

In contrast, our proposed SI-score given in Eq. (6) yields $\text{SI}(P_{d,n}, Q_{d,n}) = \sqrt{0.5}$ for the same example, reflecting the inherent uncertainty in agreement in the two responses, and correctly identifies similar responses between real data and the non-semantic synthetic data.

Figure B.1 illustrates the relation between responses and empirical estimates of the distribution. To expand on Fig. 4 in the main article, Fig. B.2 shows the top 10 principal components for each model in our study, sorted by descending SI-scores.

B.1. Form of the SI-score

When we calculate the SI-scores in Section 3.3, we do so over the multinomial distributions P_d, Q_d. To clarify, we consider the inner products and norms as being taken over the *support set*, not the multivariate dimensions.

Let $P_d, Q_d \in \mathbb{R}^N$ and let $1 \in \mathbb{R}^N$. Define the augmented vectors

$$\tilde{P}_d = \big[P_d, 1 - P_d\big], \qquad \tilde{Q}_d = \big[Q_d, 1 - Q_d\big] \in \mathbb{R}^{2N}.$$

Then we calculate the scores by averaging over the multi-

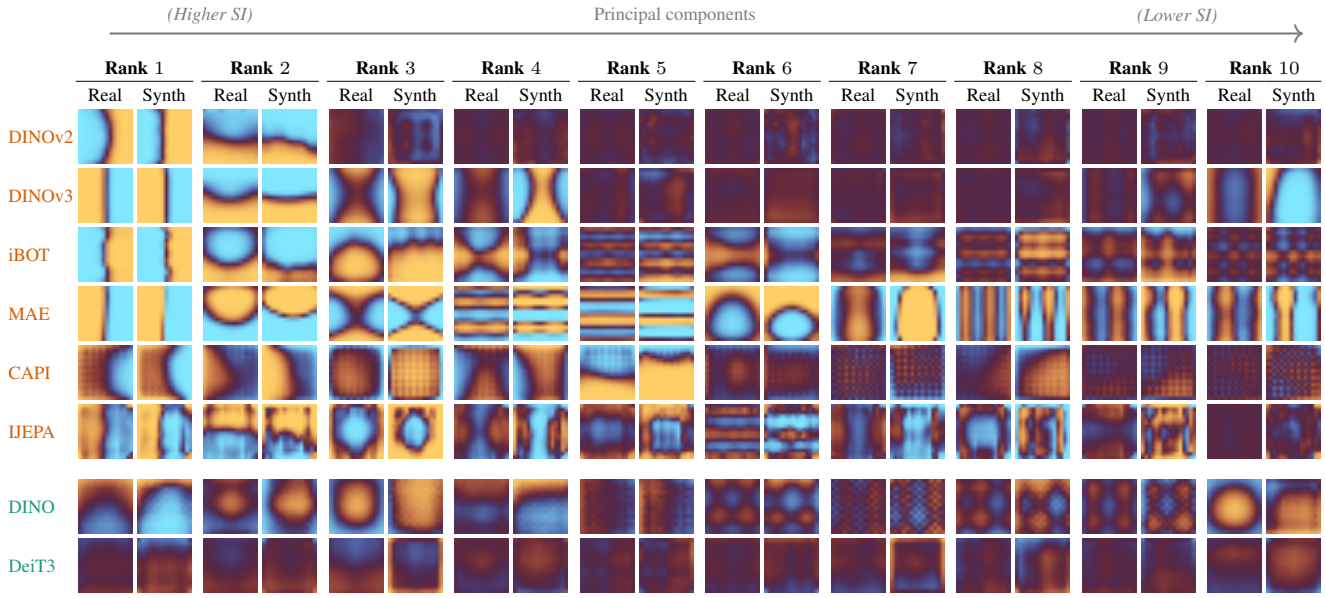

Figure B.2. Distributions for the top 10 principal components, ranked by semantic invariance (SI). **Real** columns correspond to P_d and **Synth** correspond with Q_d from Section 3.3, and each column compares the activations for real images from ImageNet, and generated synthetic images. MIM models are shown in the top rows; non-MIM models in the bottom rows. MIM models exhibit higher semantic invariance than the non-MIM models, as seen by the clear positional bias in the activations.

variate dimensions, yielding

$$s_d = \mathrm{SI}(P_d, Q_d) \tag{B.1a}$$

$$= \frac{1}{N} 1^\top \left(2 \frac{\langle \tilde{P}_d, \tilde{Q}_d \rangle}{\|\tilde{P}_d\|_2 + \|\tilde{Q}_d\|_2} \right), \tag{B.1b}$$

$$= \frac{2}{N} \sum_n \frac{P_{d,n} Q_{d,n} + (1 - P_{d,n})(1 - Q_{d,n})}{\sqrt{P_{d,n}^2 + (1 - P_{d,n})^2} + \sqrt{Q_{d,n}^2 + (1 - Q_{d,n})^2}}, \tag{B.1c}$$

where $\langle \cdot, \cdot \rangle$ and $\| \cdot \|_2$ denotes the Euclidean inner product and norm (over the support set), respectively.

C. Generating synthetic images

Let $\Omega = \{1, \ldots, H\} \times \{1, \ldots, W\}$ and $X \in \mathbb{R}^{C \times H \times W}$. For each image, draw mixture weights

$$\mathbf{w} = (w_1, w_2, w_3) \sim \mathrm{Dir}(\alpha_1, \alpha_2, \alpha_3). \tag{C.1}$$

We generate three components independently:

1. **Pink Noise**: X_{pink} is zero-mean with isotropic power spectrum

$$\mathbb{E}\big[\, |\mathcal{F}\{X_{\text{pink}}\}(\xi)|^2 \,\big] \propto \|\xi\|^{-\beta}, \qquad \xi \in \mathbb{Z}^2 \setminus \{0\},$$

with $\beta \approx 2$, following the power-law slope typical of natural images [30].

2. **Modulated White Noise**: $X_{\text{white}} = M \odot W$, where $W \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$ and the nonnegative modulation $M = g(P)$ is a smooth field obtained from a pink process P (as above) and a bounded mapping g that sets the local standard deviation. This yields a heteroscedastic Gaussian field with variance $\sigma^2(x) = M(x)^2$ and long-range variance correlations.

3. **Gradient Field**: X_{grad} is a random low-degree polynomial—or equivalently, a very low-pass random field—concentrating energy near $\xi = 0$.

Synthesized images are then given by the convex mixture

$$X = w_1 X_{\text{white}} + w_2 X_{\text{pink}} + w_3 X_{\text{grad}}. \tag{C.2}$$

Assuming zero mean and independence between components, the expected power spectrum of X is

$$S_X(\xi) = \mathbb{E}\big[\, |\mathcal{F}\{X\}(\xi)|^2 \,\big] = \sum_{i=1}^{3} \mathbb{E}[w_i^2]\, S_{X_i}(\xi), \tag{C.3}$$

i.e., a convex combination of the components' spectra.

Simoncelli and Olshausen [30] show that natural images exhibit approximate scale invariance with a $1/\|\xi\|^\beta$ law in the power spectrum (amplitude $\sim 1/\|\xi\|^{\beta/2}$), together with large-scale illumination/contrast fluctuations. In the construction above, X_{pink} directly imposes the $1/\|\xi\|^\beta$ decay, giving second-order statistics aligned with natural image ensembles. Meanwhile, X_{grad} injects additional low-frequency near-DC energy, modeling global trends and illumination. Finally, X_{white} introduces spatially varying contrast via a pink variance field, capturing long-range correlations of local variance found in natural scenes.

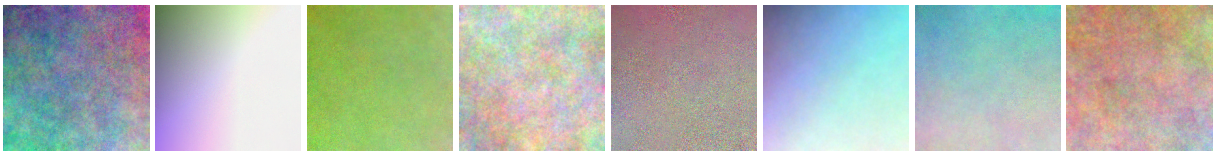

Figure C.1. Examples of generated synthetic non-semantic images.

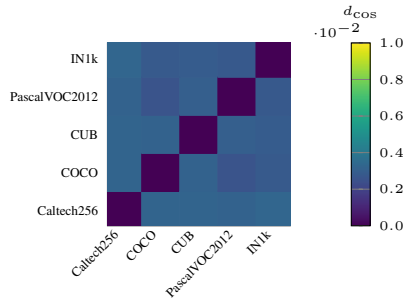

Figure D.1. Cosine distance between SI-scores for DINOv2 using various datasets for semantically informative images. The low values in $(.0025, .0032)$ indicate that the SI-score is consistent across datasets.

Thus $S_X(\xi)$ inherits a natural-image-like spectrum: a power-law falloff dominated by X_{pink}, boosted near $\xi = 0$ by X_{grad}, and with heteroscedasticity from X_{mw}. We illustrate examples of synthetic images in Fig. C.1.

During testing, we also experimented with standard probability distributions such as Gaussian or Uniform noise, in addition to simple mono-colored and gradient images, which provides similar responses to our proposed synthetic data. However, we considered these less appropriate given the mismatch in frequency response, which differs significantly from natural images. Hence, we designed our synthesis to better match key properties of natural images without additional semantic content.

D. SI-score sensitivity to dataset choice

Expanding on Sec. 4.3, we show the SI-score distance for each pair of datasets in Fig. D.1.

E. Effect of the Scaling Function

The filtering of SI scores acts as a smooth low-pass filter over the PCA spectrum. We selected the Fermi window as a smooth approximation of hard truncation, due to its precedence as a regularizer in image reconstruction. Figure E.1 illustrates the effect of the scaling function on the SI scores.

We ablate the effect of removing the scaling function in SOAP in Section 4.3, showing the downstream performance of kNN classification on the aggregated patch embeddings in Table 5. We provide additional results in Table E.1 by showing the effect for salient segmentation. We

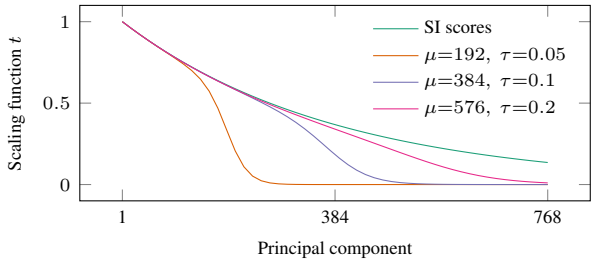

Figure E.1. Plot showing the effect of the scaling function, filtering with the Fermi window in Eq. (8), on semantic invariance (SI) scores with various choices of μ, τ. The μ parameter controls how many components are filtered, while τ controls the smoothness.

Table E.1. Ablation on the scaling function in Eq. (8), evaluated on salient segmentation for ECSSD [46]. Removing the scaling function from SOAP generally leads to reduced performance.

	SOAP without scaling			SOAP with scaling		
Pretrain	max F_β	IoU	Acc.	max F_β	IoU	Acc.
DINOv2	81.615	73.533	89.472	80.633	72.559	88.687
DINOv3	39.867	32.426	59.745	42.633	33.742	61.975
iBOT	64.702	58.401	76.419	66.557	60.167	78.340
CAPI	85.555	78.446	93.460	85.219	78.084	92.600
MAE	73.264	62.877	85.655	82.094	72.118	91.444
I-JEPA	32.646	24.327	68.927	40.239	31.162	71.406

see a boost in performance by 1–9% points for all models, except DINOv2 and CAPI, which have reduced performance by $< 1\%$. Since the majority of the SI-scores are > 0.6 for all models, projecting the representations with SOAP without scaling the SI-scores results in suppressing a majority of the principal components, reducing performance when useful information is encoded in any but the most semantic components. We chose to include the ablation with kNN in the main paper, as the salient segmentation task requires less fine-grained information about the patch contents, making kNN more informative as an ablation task.

F. Evaluation Details

Throughout the paper, we provide several evaluations and experiments. In this section, we exposition some of the details for each evaluation method.

F.1. TokenCut

We use the official TokenCut [41] implementation with their graph cut segmentation algorithm for the patch embeddings and the bilateral solver for edge-aware post-processing to

Table F.1. Ablation over the TokenCut parameter τ_{TC} on ECSSD [46] for DINOv2 and CAPI.

	DINOv2 ViT-B/16			CAPI ViT-L/14		
τ_{TC}	max F_β	IoU	Acc.	max F_β	IoU	Acc.
0.2	79.037	70.033	87.720	64.927	53.906	78.027
0.3	79.803	71.751	87.953	72.456	66.083	84.334
0.4	79.177	71.708	87.655	70.604	64.627	84.139

refine the segmentations up to original image size. TokenCut is a natural extension of spectral clustering and graph cuts to token representations, where patches are either classified as belonging to foreground or background by taking the inner product of the Fiedler vector of a filtered Gramian matrix over the tokens. Contrary to Wang et al. [41], we find that using the final output features yields better results for all models except MAE. Our reported results are thus for the out features for all models in our study, except MAE, for which we use the key features. We set $\tau_{\text{TC}} = 0.3$ for all models; Table F.1 shows this setting yield better results for DINOv2 and CAPI. Otherwise, we follow original implementation.

Selecting foreground partition using salient principal components. We also observe that some principal components have a strong center bias in the activations. This can be partially explained by object center bias in the training data. Comparing with the activation maps of synthetic input data in Fig. B.2 shows that in several cases the center bias is not present. This indicates that these components are responding to relative saliency or instance level correlations for each of the local patches, rather than an encoded positional center bias in the model.

Given a principal component v_d with salient activations, we can effectively determine which patches are likely to be part of the foreground or class level object. We find that we can improve zero-shot salient segmentation with TokenCut by selecting the foreground partition based on the patch with the highest response $\langle v_d, z \rangle$. In contrast, TokenCut selects the patch with maximum absolute value in its feature vector. The results in Table F.2 show out-of-the-box improvement across the board, where we use the strongest salient principal component to guide foreground selection for each model. We show results for all MIM models in our study, except I-JEPA which did not have a good salient principal component. We observed low performance on salient segmentation for all our experiments with I-JEPA, which suggests that the patch embeddings are not informative for this task.

F.2. kNN segmentation

We evaluate segmentation quality without any learnable parameters using a patch-level k-nearest neighbor approach. We first extract patch features from all training images using the frozen backbone and build a feature bank of ℓ_2-normalized patch embeddings paired with their ground truth labels, obtained by downsampling the segmentation masks to the patch grid via nearest-neighbor interpolation. At evaluation time, each query patch is classified by retrieving its $k = 30$ nearest neighbors from the feature bank using cosine similarity, with temperature-scaled ($\tau = 0.07$) weighted voting over the neighbor labels. This approach directly probes the spatial quality of frozen patch representations without introducing any learnable parameters, making it a good diagnostic for raw feature quality.

Table F.2. Using salient principal components to select foreground partition in TokenCut. Results are shows for corrected embeddings after cleaning with SOAP.

		Max. abs. val.			Max. Sal. PC reponse		
Model	Arch.	max F_β	IoU	Acc.	max F_β	IoU	Acc.
DINOv2	ViT-B/14	71.461	64.129	83.292	80.633	72.559	88.687
DINOv3	ViT-B/16	33.360	25.111	46.521	42.633	33.742	61.975
iBOT	ViT-B/16	64.432	57.978	80.093	66.557	60.167	78.340
CAPI	ViT-L/14	74.506	68.045	86.3234	85.219	78.084	92.600
MAE	ViT-B/16	80.525	70.705	90.506	82.094	72.118	91.444
Franca	ViT-B/14	77.660	70.830	87.577	84.176	76.985	91.514

In Tab. F.3 we provide additional results for PascalVOC [18], expanding the results for ADE20k [51, 52] that were reported in Tab. 3 in the main paper.

F.3. kNN classification

In this section we provide details on our kNN classification evaluation protocol. In the main article we report kNN classification by attention- and entropy-weighted aggregation of patch predictions; the exact details are in Sec. F.3.1. Furthermore, we report kNN classification on the averaged patch embeddings in Sec. F.3.2 for completeness.

F.3.1. kNN classification by weighted aggregation of patch predictions

In Tab. 4 in Sec. 4.2, we evaluate frozen patch features on ImageNet [17] using a patch-level k-nearest neighbor classification protocol with $k = 20$ and temperature $\tau = 0.07$. We extract patch tokens from the last layer of each frozen backbone and reduce their dimensionality to 256 using PCA. Features are ℓ_2-normalized and stored in a feature bank sharded across 2 GPUs in float16 precision. Each patch in a query image retrieves its k nearest neighbors from the training feature bank via cosine similarity, producing per-patch class probability distributions through temperature-scaled softmax weighting. The per-patch probabilities are then aggregated into a single image-level prediction using one of two weighting schemes:

1. *CLS attention weighting*, which uses the attention weights from the `[CLS]` token in the last self-attention layer of the backbone to weight each patch's contribution, thus leveraging the model's own learned notion of patch importance.

Table F.3. kNN segmentation comparison of original vs. SOAP-corrected embeddings across backbones, reporting mean IoU and pixel accuracy (Acc) for the ADE20 [51, 52] and PascalVOC [18] benchmarks.

		ADE20k				PascalVOC			
		Original emb.		Corrected emb.		Original emb.		Corrected emb.	
Pretrain	Model	IoU	Acc	IoU	Acc	IoU	Acc	IoU	Acc
DINOv2	ViT-B16	40.25	74.60	40.81 ↑0.56	74.72 ↑0.12	73.34	93.74	74.07 ↑0.73	93.90 ↑0.16
DINOv3	ViT-B16	43.85	77.94	44.58 ↑0.73	78.10 ↑0.16	78.46	95.41	79.43 ↑0.97	95.58 ↑0.17
iBOT	ViT-B16	27.73	70.86	28.43 ↑0.70	71.27 ↑0.40	61.10	91.25	62.16 ↑1.06	91.50 ↑0.25
CAPI	ViT-L14	31.38	71.63	31.64 ↑0.26	71.77 ↑0.14	60.40	91.40	60.80 ↑0.40	91.50 ↑0.09
MAE	ViT-B16	11.88	58.00	12.65 ↑0.77	58.60 ↑0.59	27.99	82.624	30.39 ↑2.37	83.27 ↑0.65
I-JEPA	ViT-H14	20.95	60.27	21.26 ↑0.31	60.29 ↑0.01	57.77	89.22	58.50 ↑0.73	89.34 ↑0.13
DINO	ViT-B16	21.21	66.49	21.21 0.00	66.49 0.00	47.32	88.01	47.32 0.00	88.01 0.00

2. *Entropy weighting*, which assigns higher weight to patches whose k-NN probability distributions have lower entropy, favoring patches that yield more confident predictions.

We aggregate by CLS attention weighting for models trained with an instance discrimination objective (DINO, DINOv2, DINOv3, iBOT), as these methods explicitly train the `[CLS]` token to capture global image semantics through their contrastive or self-distillation losses, yielding meaningful attention distributions over patches. For models trained exclusively with a masked image modeling objective (I-JEPA, MAE, CAPI), the `[CLS]` token is either absent or not trained to aggregate global information, so its attention weights are not informative for patch weighting. We therefore use entropy-based aggregation for these models, which is agnostic to the pretraining objective and instead relies on the confidence of the per-patch k-NN predictions themselves.

In Tab. F.4 we provide additional results for iNat2018 [34], expanding the results for ImageNet [17] that were reported in Tab. 4 in the main paper.

F.3.2. kNN classification on averaged patch embeddings.

We additionally perform kNN classification on ImageNet [17] by average pooling the patch embeddings, and matching the validation embeddings to the k nearest embeddings from the training set. We follow the kNN evaluation script by Caron et al. [11], and set $k = 20$ number of neighbors and 0.07 temperature for the voting coefficient. We compare the top-1 and top-5 accuracies of the average of the patch embeddings, and the corrected patch embeddings. The results in Table F.5 show only modest improvements after correcting for invariant components. This is the expected result, as instance tasks are not as reliant on local semantics, and the benefit of SOAP may be dampened by the uniform aggregation, as spatial information about the locality of patches reduces due to averaging them out.

G. Linear evaluation protocols

Our intention with restricting evaluation to zero-shot protocols (TokenCut, kNN) is to measure the *intrinsic fidelity* of the representations, while *limiting confounding factors*. As SOAP uses linear PCA, a learnable head can adapt to suppress positional noise when this information is unhelpful for the task. Linear evaluation protocols are thus unsuitable for diagnosing positional noise, as they can adapt, unintentionally masking the issue. Indeed, linear probing, attentive probing, and linear segmentation yield similar results with SOAP; see Tab. G.1, Tab. G.2, and Tab. G.3. The difference in performance with and without SOAP is very low—this level of variation in performance is expected when probing with learnable heads, and the difference is thus too small to attribute any change in performance to SOAP. We describe each of these evaluation protocols in detail below.

In contrast, kNN directly probes representation geometry without learned transformations, making it a faithful diagnostic for raw feature quality [11, 44]. This matters in practice: using SSL representations out-of-the-box is common, and practitioners unaware of positional noise may encounter false positives from patch embeddings at similar relative locations.

G.1. Linear evaluation.

We perform linear evaluation on the averaged patch embeddings of the last output layer of each model on ImageNet [17]. We follow the standard protocol of training a single linear layer for classification on top of the frozen features for 100 epochs. We use a standard stochastic gradient descent optimizer (`SGD`) with a base learning rate of 0.001, momentum 0.9, cosine learning rate decay, and a batch size of 256 with 4 GPUs (effective batch size 1024). Following protocol, the learning rate is scaled by

$$\text{lr} = \frac{\text{base lr} \times \text{batch size} \times \text{num GPUs}}{256}.$$

The results in Tab. G.1 show minor changes in performance when the embeddings are corrected with SOAP, reflecting that suppressing positional noise has little effect when the classification head is learnable.

Table F.4. Weighted kNN classification by aggregating patch prediction on iNat2018 [34] and ImageNet [17], expanding results from Tab. 4. We compare original vs. SOAP-corrected embeddings across backbones.

		ImageNet				iNat			
		Original emb.		Corrected emb.		Original emb.		Corrected emb.	
Pretrain	Model	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
DINOv2	ViT-B16	82.32	96.29	82.59 ↑0.27	96.30 ↑0.01	58.80	82.99	60.36 ↑1.56	83.75 ↑0.76
DINOv3	ViT-B16	81.47	95.57	81.48 ↑0.01	95.59 ↑0.01	58.25	79.52	58.76 ↑0.51	80.12 ↑0.60
iBOT	ViT-B16	71.45	90.03	71.60 ↑0.15	90.10 ↑0.07	27.64	49.06	27.70 ↑0.06	49.73 ↑0.66
CAPI†	ViT-L14	70.81	91.09	71.25 ↑0.43	91.33 ↑0.24	26.88	51.18	27.64 ↑0.76	52.25 ↑1.07
MAE†	ViT-B16	59.30	81.49	60.62 ↑1.32	82.38 ↑0.89	16.43	31.36	19.12 ↑2.70	36.61 ↑5.25
I-JEPA†	ViT-H14	75.68	91.70	75.89 ↑0.21	91.76 ↑0.06	17.29	35.34	18.01 ↑0.73	36.06 ↑0.72
DINO	ViT-B16	66.08	86.13	66.08 0.00	86.13 0.00	24.64	44.60	24.64 0.00	44.60 0.00

†Aggregation weighted by entropy for models with no class token objective; otherwise weighted by class attention.

Table F.5. kNN classification of average pooled patch embeddings on ImageNet [17]. We compare the original embeddings with the SOAP-corrected embeddings for each backbone, and report top-1 and top-5 accuracies.

		Original embeddings		Corrected embeddings	
Pretrain	Model	Acc@1	Acc@5	Acc@1	Acc@5
DINOv2	ViT-B16	77.064	91.624	77.100 ↑0.036	91.636 ↑0.012
DINOv3	ViT-B16	76.542	91.530	76.588 ↑0.046	91.612 ↑0.082
iBOT	ViT-B16	59.170	79.612	59.498 ↑0.328	79.918 ↑0.306
CAPI	ViT-L14	56.250	77.490	56.444 ↑0.194	77.742 ↑0.252
MAE	ViT-B16	47.488	69.168	47.758 ↑0.27	69.442 ↑0.274
I-JEPA	ViT-H14	71.382	86.144	71.390 ↑0.008	86.168 ↑0.024
DINO	ViT-B16	55.216	75.740	55.216 0.000	75.740 0.000

G.2. Attentive probing.

The attentive probing protocol replaces the global average pooling of patch tokens with a learnable attention mechanism that computes a weighted aggregation over the patch tokens before classification [6]. Specifically, a lightweight two-layer MLP computes per-patch attention logits, which are normalized via softmax to produce attention weights over the spatial positions. The attended feature is then passed to a linear classifier. This allows the probe to selectively focus on the most informative patches, which is particularly beneficial for methods where discriminative information is distributed across patch tokens rather than concentrated in a single global representation. We otherwise follow the linear evaluation protocol, training for 100 epochs with `SGD`, a base learning rate of 0.0025, momentum 0.9, cosine learning rate decay, and a batch size of 256 with 4 GPUs (effective batch size 1024). The results in Tab. G.2 show minor changes in performance when the embeddings are corrected with SOAP, reflecting that suppressing positional noise has little effect when the classification head is learnable.

G.3. Linear Segmentation.

We evaluate segmentation quality by training a linear segmentation head on top of frozen patch features on ADE20k [51, 52]. The segmentation head consists of a single 1×1 convolution applied to the spatial patch feature map, followed by bilinear upsampling to the original image resolution. We train with cross-entropy loss and `SGD` with momentum 0.9, polynomial learning rate decay with power 0.9, and a batch size of 32 per GPU across 4 GPUs (effective batch size 128) for 80 epochs. The learning rate is selected from $\{0.08, 0.04, 0.008\}$ based on validation mIoU for the baseline results of the original embeddings for each model. The crop size is set to be divisible by the patch size of the backbone: 512 for patch size 16 and 518 for patch size 14. Training images are augmented with random scaling (ratio 0.5–2.0×), random cropping, and random horizontal flipping; validation images are resized to the crop size. We report mean intersection over union (IoU) and per pixel accuracy in Tab. G.3, once again showing that learnable evaluation heads confound the effect of suppressing positional noise with SOAP.

Table G.1. Linear evaluation protocol.

		Original embeddings		Corrected embeddings	
Pretrain	Model	Acc@1	Acc@5	Acc@1	Acc@5
DINOv2	ViT-B16	81.13	96.01	81.15 ↑0.02	96.01 0.00
DINOv3	ViT-B16	76.77	93.89	76.76 ↓-0.01	93.90 ↑0.01
iBOT	ViT-B16	72.89	91.37	72.93 ↑0.04	91.35 ↓-0.02
CAPI	ViT-L14	63.09	85.41	63.17 ↑0.08	85.46 ↑0.05
MAE	ViT-B16	50.60	74.83	50.63 ↑0.03	74.87 ↑0.04
I-JEPA	ViT-H14	74.99	90.65	74.99 0.00	90.69 ↑0.04
DINO	ViT-B16	66.36	86.43	66.39 ↑0.03	86.41 ↓-0.02

Table G.2. Attentive probing.

		Original embeddings		Corrected embeddings	
Pretrain	Model	Acc@1	Acc@5	Acc@1	Acc@5
DINOv2	ViT-B16	84.97	97.20	85.00 ↑0.03	97.22 ↑0.02
DINOv3	ViT-B16	83.40	96.56	83.46 ↑0.06	96.61 ↑0.05
iBOT	ViT-B16	79.05	94.34	79.04 ↓-0.01	94.31 ↓-0.03
CAPI	ViT-L14	81.75	95.84	81.72 ↓-0.03	95.93 ↑0.09
MAE	ViT-B16	67.80	87.44	67.90 ↑0.10	87.47 ↑0.03
I-JEPA	ViT-H14	77.66	92.84	77.60 ↓-0.06	92.77 ↓-0.07
DINO	ViT-B16	72.56	90.58	72.50 ↓-0.06	90.58 0.00

Table G.3. Linear segmentation.

		Original embeddings		Corrected embeddings	
Pretrain	Model	IoU	Acc	IoU	Acc
DINOv2	ViT-B16	47.54	80.21	47.53 ↓ -0.01	80.20 ↓ -0.01
DINOv3	ViT-B16	49.35	82.31	49.28 ↓ -0.07	82.25 ↓ -0.06
iBOT	ViT-B16	35.61	75.97	35.61 ↑ 0.00	76.12 ↑ 0.15
CAPI	ViT-L14	41.96	78.73	41.98 ↑ 0.02	78.55 ↓ -0.18
MAE	ViT-B16	20.43	65.10	20.52 ↑ 0.09	65.48 ↑ 0.38
I-JEPA	ViT-H14	27.43	68.71	27.53 ↑ 0.10	68.77 ↑ 0.06
DINO	ViT-B16	28.95	71.28	28.88 ↓ -0.07	71.19 ↓ -0.09

H. Additional salient segmentation examples

We show additional examples of salient segmentation using TokenCut [41] in Figs. H.1 and H.2, for the DUTS [38] and DUTOMRON [47] datasets, respectively. The images displayed are from the first samples in the datasets, and were not cherry picked except to show different outcomes of using SOAP in the case of DUTOMRON. We show both the coarse per-patch prediction maps, and the refined maps after using the bilateral solver for edge aware post-processing; see App. F.1 for more TokenCut details.

Figure H.1. Examples of salient segmentation from DUTS [38] using TokenCut [41] with frozen CAPI [16] on the raw embeddings (Raw pred.) and after correcting with SOAP (SOAP pred.). We show the predictions per patch and after refining the segmentation maps with the bilateral solver (BS). Suppressing positional noise with SOAP either matches or improves the zero-shot saliency maps. These examples are from the first samples in the dataset.

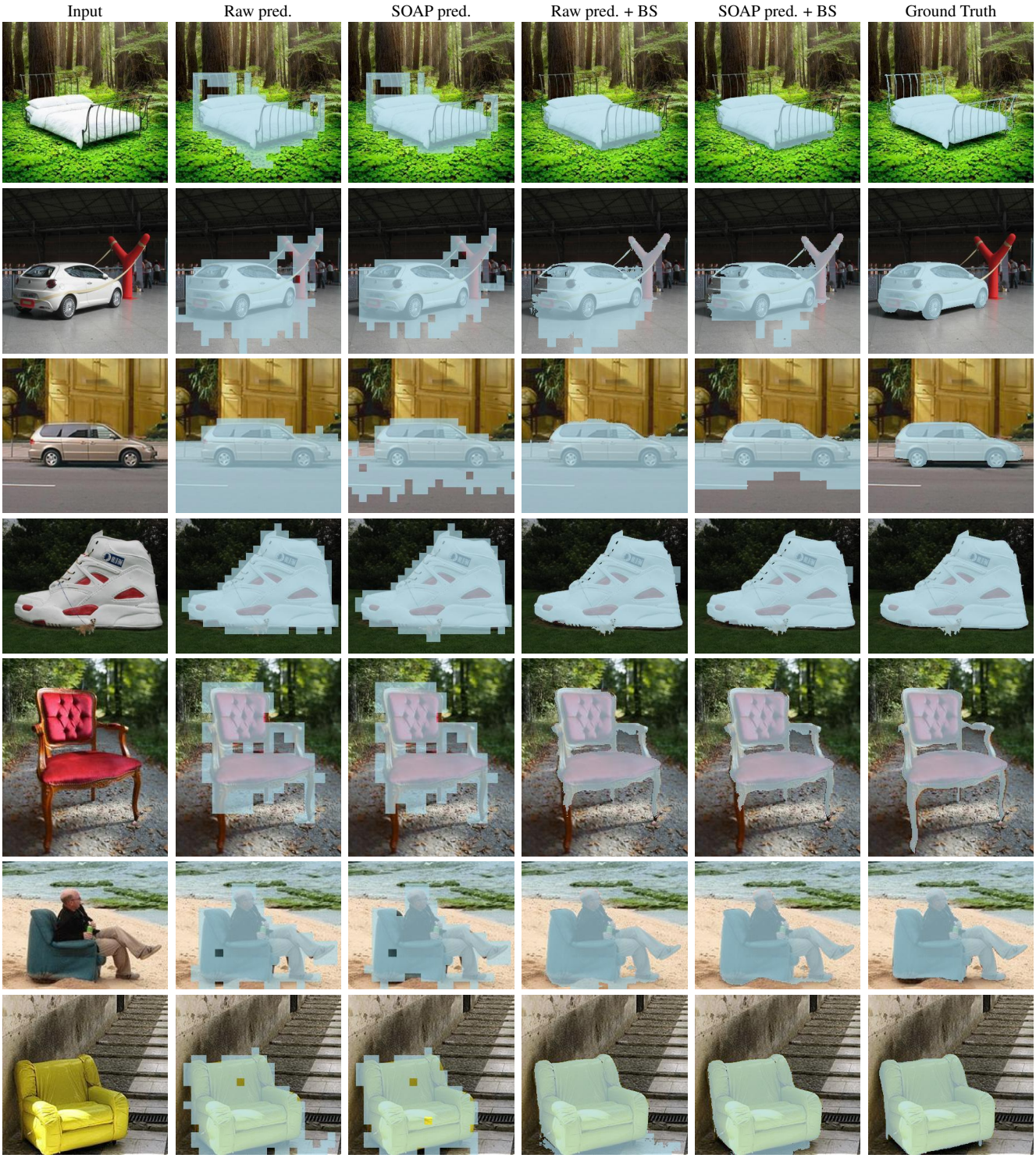

Figure H.2. Examples of salient segmentation from DUTOMRON [47] using TokenCut [41] with frozen CAPI [16] on the raw embeddings (Raw pred.) and after correcting with SOAP (SOAP pred.). Suppressing positional noise with SOAP either matches or improves the zero-shot saliency maps. We show the predictions per patch and after refining the segmentation maps with the bilateral solver (BS). These examples are from the first samples in the dataset, and were not cherry picked except to show different outcomes of using SOAP.